Lab4

Ahmed Alhasan, Yash Pawar, Mohsen Pirmoradiyan

2/17/2020

Question 1: Computations with Metropolis-Hastings

The target probability density function:

$$f(x) \propto x^5 exp(-x), \quad x > 0$$

Our goal is to implement Metropolis-Hastings algorithm to generate samples from this distribution by using two different proposal distribution and compare the results. The function of target is:

```
target <- function(x){
  return(x**5 * exp(-x))
}
```

1) log-normal as the proposal distribution

The proposal distribution is log-normal distribution $LN(x_t, 1)$. We use the R standard function for log-normal distribution: $rlnorm()$ for random number generator and $dlnorm()$ to compute the density of x given a mean value.

Using this proposal, we now implement the MH algorithm for generating the distribution from $f(x)$. The steps are as follow:

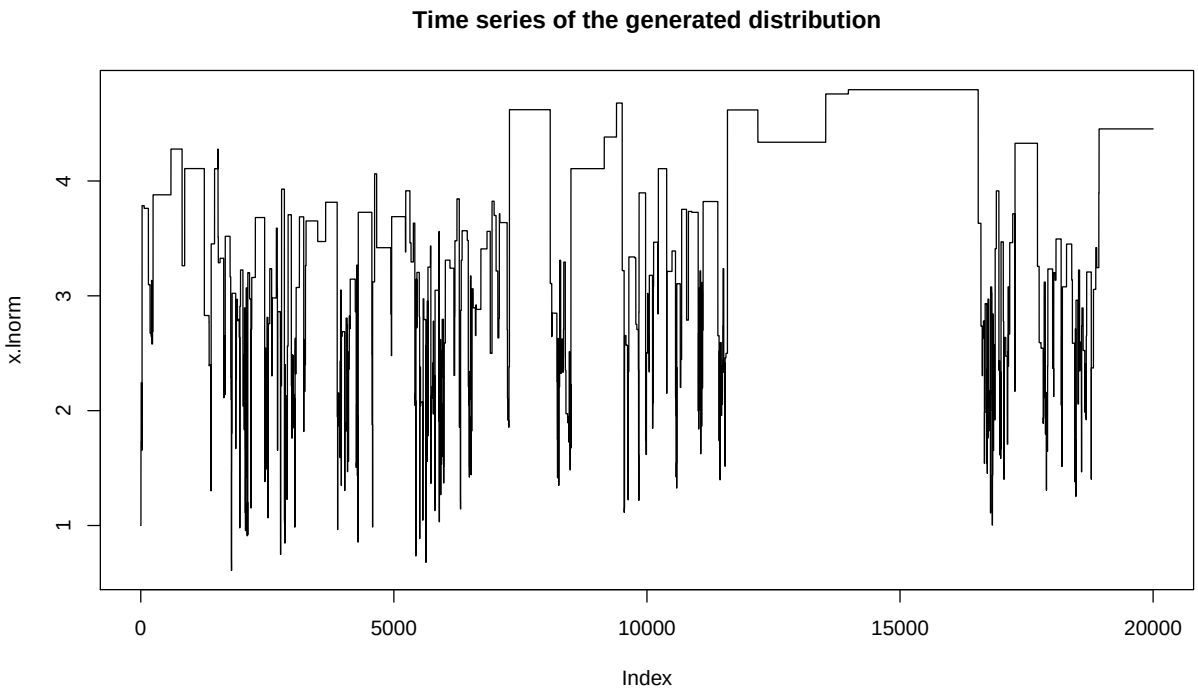
1) First we initialize the algorithm by a starting point, x = 1
2) Using this point, as current x, we move randomly to a nearby point using the log-normal proposal distribution.
3) Compute the acceptance probability:

$$A = min(1, \frac{f(x.proposed)q(x.current|x.proposed)}{f(x.current)q(x.proposed|x.current)})$$

4) Generate a random uniform number$U(0, 1)$; if this number is less than A we accept the proposed x as the next point; otherwise we stay, and repeat the process

```
lnorm.HM = function(n, start.point){

  x = rep(0, 20000)
  x[1] = 1
  for (i in 2:length(x)) {
    x_cur = x[i-1]
    x_propos = rlnorm(1, meanlog = x_cur)
    A = min(1,(target(x_propos)*dlnorm(x_cur, meanlog = x_propos))/(target(x_cur)*dlnorm(x_propos, meanl

    if(runif(1) < A){
      x[i] = x_propos}else{
        x[i] = x[i-1]

      }
  }
```

```
  return(x)

}

set.seed(12345)
x.lnorm = lnorm.HM(10000, 1)
```



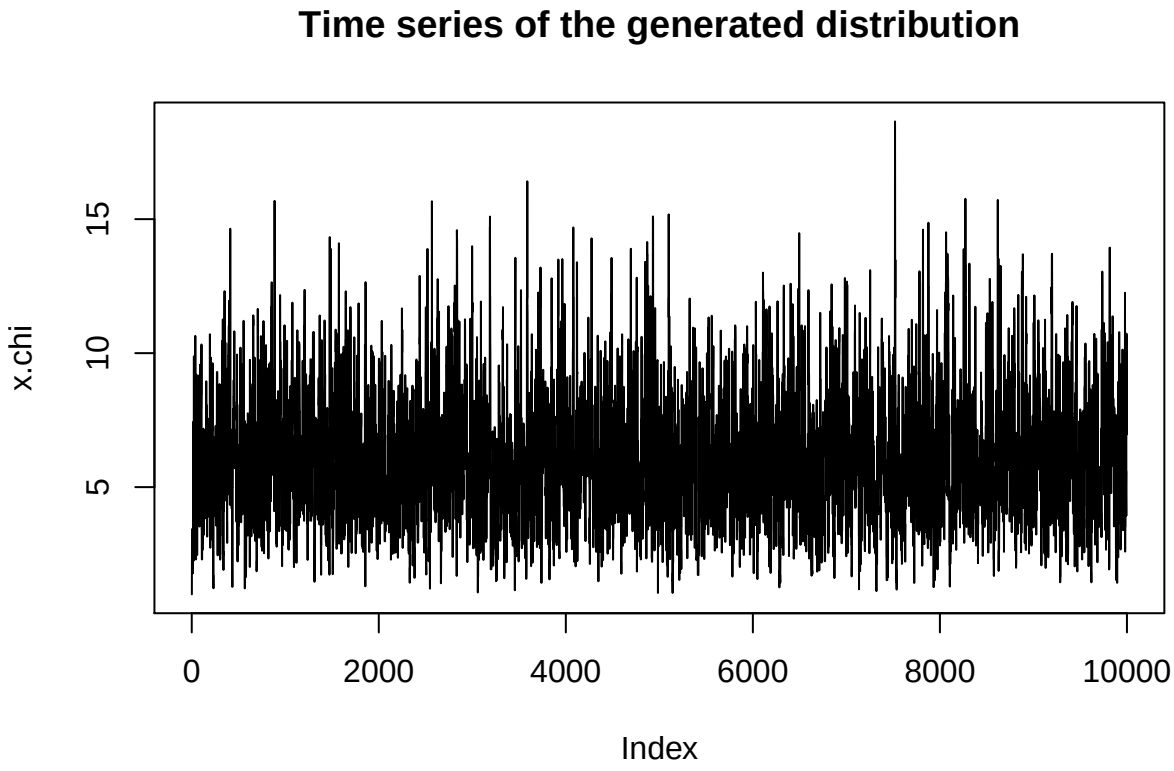
Depening the starting point, it will take the sampling some time to converge which is reffered to as “burn-in”. In other words, “burn-in” is the time needed for Markov Chain method to reach its equilibrium distribution. The figure above does not show convergence. Some flat parts can be seen that suggest that the method could not find a more likely point to move in. High jumps over the domain are obvious. Therefore it seems that the iteration did not converge and as a result burn-in may not be applicable here.

You may say that there is no burn-in period, but it's obvious not "as a result" of the jumps.

2) The chi-square distribution $\chi^2(\lfloor X_t + 1 \rfloor)$ as a proposal distribution

For this part we use chi-square distribution as the proposal distribution. We implement the R standard functions for chi-square distribution(*rchisq*() *and dchisq*())

```
chi.HM = function(n, start.point){

  x = rep(0, n)
  x[1] = start.point
  for (i in 2:length(x)) {
    x_cur = x[i-1]
    x_propos = rchisq(1, df=floor(x_cur+1))
    A = min(1,(target(x_propos)*dchisq(x_cur,floor(x_propos+1)))/(target(x_cur)*dchisq(x_propos,floor(x
    if(runif(1) < A){
      x[i] = x_propos}else{
        x[i] = x[i-1]
      }
  }
  return(x)
}

set.seed(12345)
x.chi = chi.HM(10000, 1)
```



Good

Regardless of some high jumps during the sampling, it seems that in this case the sampling did converge at the beginning. It fluctuates around a mean value. Burn-in in this case may be the first few points, as it started to converge very fast.

3. Compare the results of Steps 1 and 2 and make conclusions.

Comparing the time series from step 1 and step 2 we can conclude that the chi-square was a better choice to be implemented as a proposal distribution since the convergence could be distinguished more obviously than that of log-normal distribution. Irregular patterns in log-normal result suggest lack of convergence. In contrast regularity in sampling using chi-square could be reasoned for convergence.

4. Generate 10 MCMC sequences using the generator from Step 2 and starting points 1-10. Use the Gelman-Rubin method to analyze convergence of these sequences.

The Gelman_Rubin Covergence diagnostic evaluates the convergence by analyzing the difference between multiple Morkov chains.

Suppose we have estimated k sequences(k chains with different start points) of length n.

In this method the within-chain variance for each sequence is estimated, denoted by S_i where i denotes the i^{th} chain, from which the average within variance(W) is calculated: $W = \frac{1}{k}\sum_{i=1}^{k} {s_i}^2$.

The between-chains variances are estimated as well, denoted by B, which simply is a measure of differences between the different sequences(chains) having different start point.

Then, overall variance is estimated:

$$V = \frac{n-1}{n}W + \frac{1}{n}B$$

Finally the Gelman-Rubin factor is calculated as:

$$\sqrt{R} = \sqrt{\frac{V}{W}}$$

The values much larger than 1 indicate lack of convergence.

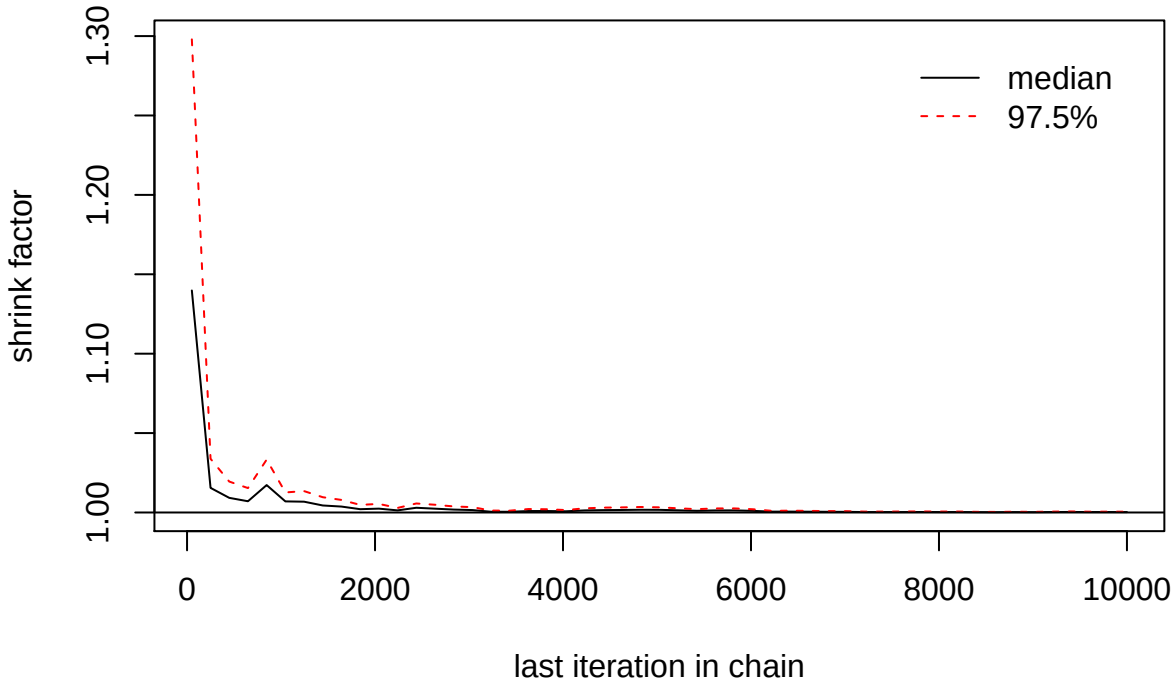
For computing this factor we used the function *gelman.diag*() from the package *coda*.

```
set.seed(12345)
Data = mcmc.list()

for (i in 1:10) {
  Data[[i]] <- as.mcmc(chi.HM(10000, i))

}
print(gelman.diag(Data))
```

Good

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

```
gelman.plot(Data)
```



Good

As the results show, the Gelman-Rubin factor for our 10 chains and the iteration of 10000 is 1 which indicates the good convergence.

5) Estimete

$$\int_0^\infty x f(x) dx$$

5.1) Using samples from step 1 Given f(x) as the density function, this integration estimates the expected value of $f(x)$ which is the mean of sample over the generated series which is:

$$\frac{1}{n}\sum_{i=1}^{n} x.lnorm_i$$

This notation is confusing. Just writing \frac{1}{n} \sum_i x_i is enough

```
mean(x.lnorm)
```

```
## [1] 3.748894
```

5.2) Using samples from step 2 Given f(x) as the density function, this integration estimates the expected value of $f(x)$ which is the mean of sample over the generated series which is:

$$\frac{1}{n}\sum_{i=1}^{n} x.chi_i$$

```
mean(x.chi)
```

```
## [1] 6.000143
```

6) The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

The Gamma distribution is as follows:

$$Gamma(\alpha, \beta) = f(x|\alpha, \beta) = [\frac{1}{\Gamma(\alpha)\beta^\alpha}]x^{\alpha-1}exp(-x/\beta)$$

where α and β are called "shape factor" and "rate factor" respectively. The expected value of a Gamma distribution is the multiplication of these two factors: $\alpha\beta$.

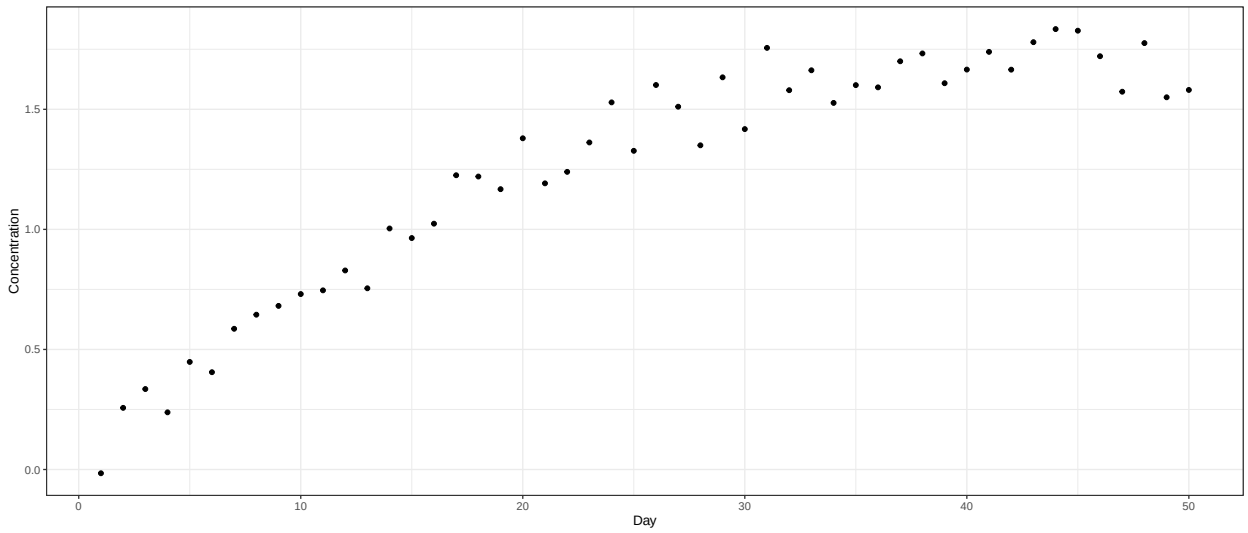
In our case $\alpha = 6$ and $\beta = 1$ which yields the expected value of 6 for $f(x)$. Our estimation from step 2 is very close to this value which indicates that the chi-square distribution was a good proposal function for sampling from this distribution.

Correct

Question 2: Gibbs sampling

1. Import the data to R and plot the dependence of Y on X.

```
chemic=load("E:/LiU/2nd Semester/Computational Statistics/Labs/4/chemical.RData")
chemic = data.frame(Day = X, Concentration = Y)

ggplot(chemic, aes(Day, Concentration))+geom_point()+theme_bw()
```



As the figure suggests a 2-degree polynomial or a quadratic function seems to well fit data.

2) Implementing the random-walk Bayesian model, present the formulae showing the likelihood $p(\vec{Y}|\vec{\mu})$ and the prior $p(\vec{\mu})$.

$$n = number\ of\ observations$$

$$\vec{\mu} = (\mu_1,\ \mu_2, ...,\ \mu_n) :\ unkonown\ parametesr$$

$$Y_i = N(\mu_i,\ variance = 0.2), \quad i = 1, ..., n$$

Where the prior is:

$$p(\mu_1) = 1$$

$$p(\mu_{i+1}|\mu_i) = N(\mu_i,\ 0.2), \quad i = 1, ..., n-1$$

The likelihood for $p(\vec{Y}|\vec{\mu})$ may be derived as:

$$L = \prod_{i=1}^{n} p(Y_i|\mu_i) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(Y_i - \mu_i)^2}{2\sigma^2})$$

$$= (\frac{1}{\sigma\sqrt{2\pi}})^n \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu_i)^2), \qquad (1)$$

Correct likelihood

The prior function $p(\vec{\mu})$ may be derived as:

$$p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)...p(\mu_n|\mu_{n-1})$$

Given $p(\mu_1) = 1$, we can conclude that:

$$p(\vec{\mu}) = \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i)$$

Correct

We know that $p(\mu_{i+1}|\mu_i) = N(\mu_i,\ 0.2), \quad i = 1, ..., n-1$, so:

$$p(\vec{\mu}) = \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i) = \prod_{i=1}^{n-1} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(\mu_{i+1} - \mu_i)^2}{2\sigma^2})$$

$$= (\frac{1}{\sigma\sqrt{2\pi}})^{n-1} \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2), \qquad (2)$$

Correct prior

(1) is the likelihood function for $p(\vec{Y}|\vec{\mu})$, and (2) is the prior function.

3. Use Bayes' Theorem to get the posterior up to a constant proportionality, and then find out the distributions of $(\mu_i|\vec{\mu}_{-i},\ \vec{Y})$

The Baye's theorem:

$$p(\theta|Y) \propto p(Y|\theta)p(\theta) = Likelihood\ *prior$$

where θ is the parameter of interest.

The parameter of interest in this problem is $\vec{\mu}$. So we are interested to derive a conditional distribution (posterior) of $p(\vec{\mu}|Y)$. The likelihood and the prior we derived in previous step may be used to find the proportionality for the posterior;$posterior \propto\ (1)\ *\ (2)$:

Correct

$$p(\vec{\mu}|Y) \propto \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu_i)^2)\ .\ \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2)$$

Correct

$$\propto \exp(-\frac{1}{2\sigma^2}[(\mu_1-Y_1)^2+\sum_{i=2}^{n}(\mu_i-Y_i)^2+(\mu_i-\mu_{i-1})^2)], \qquad (3)$$

(3) is the general form of proportionality for the posterior. To drive the conditional distribution for each μ, we first find $p(\mu_1|\mu_{-1,Y})$ considering that the proportionality principle allows us to remove the terms which do not involve any functional dependence of μ which means that we are allowed to exclude the terms which do not contain μ:

$$p(\mu_1|\mu_{-1},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(\mu_1-Y_1)^2+(\mu_1-\mu_2)^2])$$

Using Hint B the right hand sid is proportional to:

$$\exp(-\frac{1}{\sigma^2}(\mu_1-\frac{Y_1+\mu_2}{2})^2)$$

Hence,

$$p(\mu_1|\mu_{-1},\vec{Y}) \propto \exp(-\frac{1}{\sigma^2}(\mu_1-\frac{Y_1+\mu_2}{2})^2)\ \sim\ N(\frac{Y_1+\mu_2}{2},\ \frac{\sigma^2}{2}), \qquad (4)$$ Correct

Next we derive the distribution $p(\mu_n|\vec{\mu}_{-n},\vec{Y})$:

Again according to proportionality principle and the hint B we have:

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(\mu_n-Y_n)^2+(\mu_n-\mu_{n-1})^2)]$$

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto \exp(-\frac{1}{\sigma^2}(\mu_n-\frac{Y_n+\mu_{n-1}}{2})^2)\ \sim\ N(\frac{Y_n+\mu_{n-1}}{2},\ \frac{\sigma^2}{2}), \qquad (5)$$ Correct

And finally we find the conditional distribution $p(\mu_i|\vec{\mu}_{-i},\vec{Y})$ for $i=2,...,n-1$.

For the i^{th} μ there are three terms in the posterior proportionality derived as (3) which contain μ_i as there are mu_{i-1} and μ_{i+1} terms befor and after that. Hence we have:

$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2])$$

Using hint C we conclude that

$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto \exp(-\frac{3}{2\sigma^2}(\mu_i-\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3})^2)\ \sim\ N(\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3},\ \frac{\sigma^2}{3}), \qquad (6)$$ Correct

4) Gibbs sampler

We are asked to obtain 1000 vectors, $\vec{\mu}$, each of which contains $(\mu_1, \mu_2, ..., \mu_n)$, then compute the expected value of each vector and plot the resulted mean vector as the expected values of concentration.

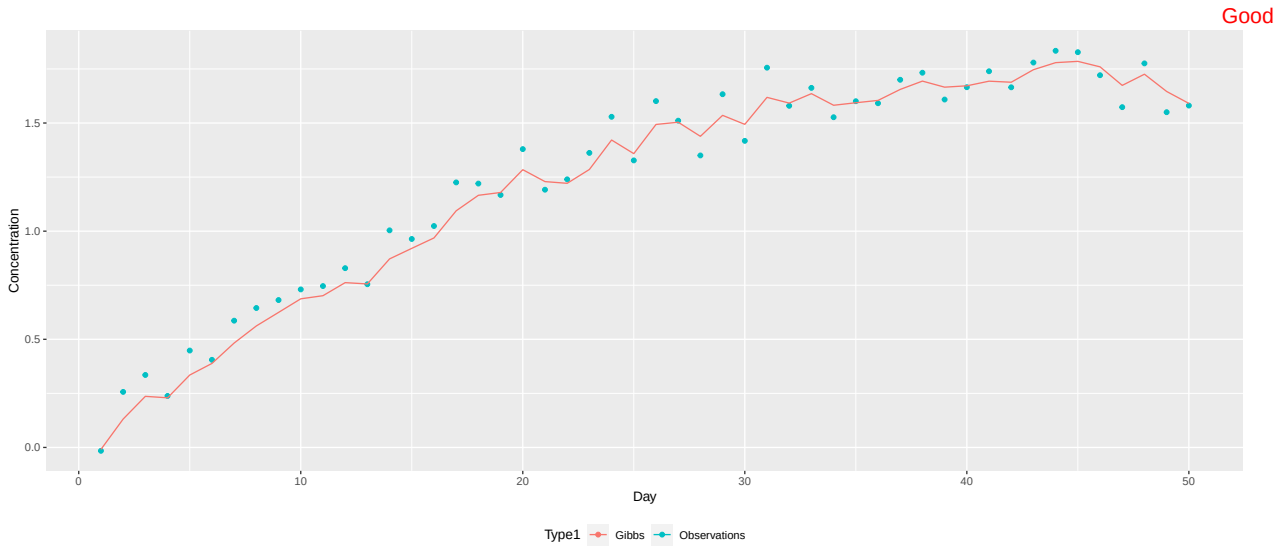
We construct a $1000 * 50$ zero matrix as the initial point. The we update each row implementing the equations $(4), (5), and (6)$ depending on the which column(which elemnt of the vectoe μ) we are updating:

```
y = chemic$Concentration
n= nrow(chemic)

M = 1000
Var = 0.2

mu = matrix(0, nrow = M, ncol = n)
set.seed(12345)
for (i in 1:M) {

  for (j in 1:n) {
    if(j ==1){
      mu[i,j] = rnorm(1, mean = (y[1]+mu[i,2])/2, sd = sqrt(Var/2))
      }else if(j == n){
        mu[i,j] = rnorm(1, mean = (y[n] + mu[i,n-1])/2, sd = sqrt(Var/2))
        }else{
          mu[i,j] = rnorm(1, mean = (y[j] + mu[i,j-1] + mu[i,j+1])/2,
                          sd = sqrt(Var/3))
        }
  }
}
```

Having computed all 1000 $\vec{\mu}$, we now compute the expected value of each vector and plot the resulted mean vector vs. day.

```
Mu.mean = apply(mu, 2, mean)
chemic$Mu = Mu.mean

chemic$Type1 = "Observations"
chemic$Type2 = "Gibbs"


ggplot(chemic)+ geom_point(mapping = aes(Day, Concentration, color=Type1))+
  geom_line(mapping = aes(Day, Mu, color = Type2)) + theme(legend.position = "bottom")
```

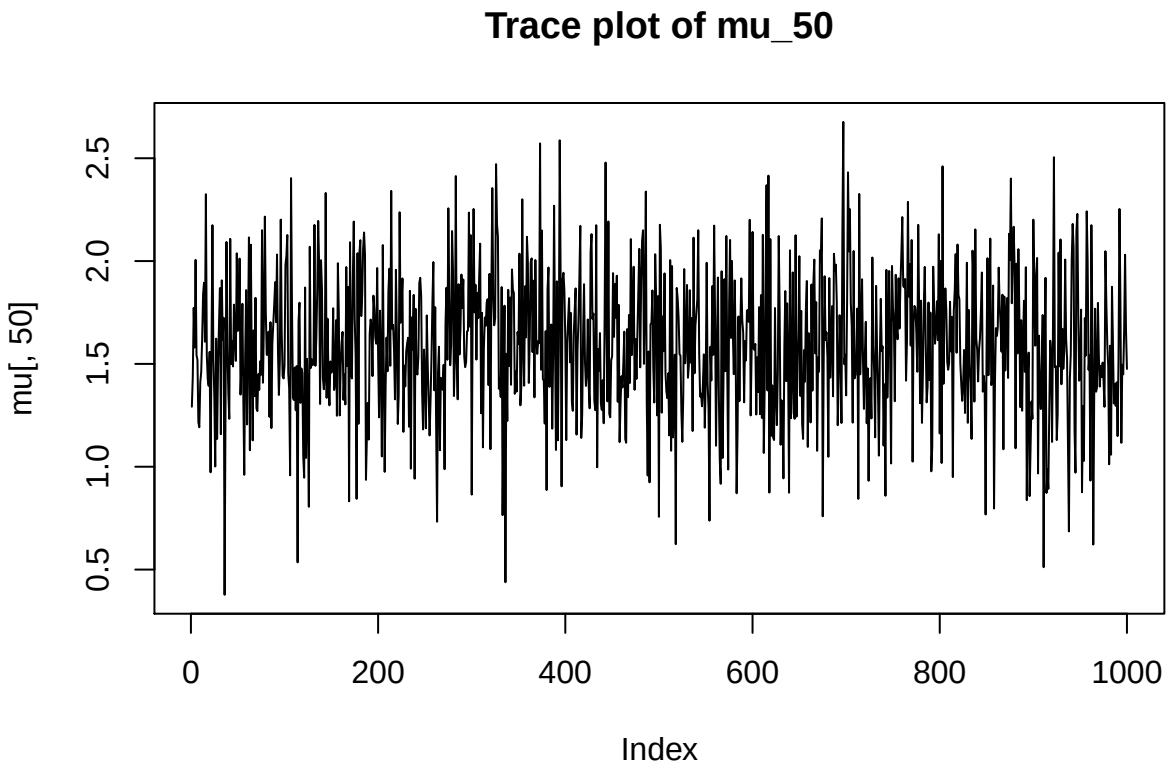
Good



As the graph shows that the sampler seems to capture the relationship between Day and Concentration. Obviously it could remove the noise.

5) Make a trace plot for µn and comment on the burn{in period and convergence.

```
plot(mu[,50], type = 'l', main = "Trace plot of mu_50")
```



The trace plot of μ_{50} can be seen in the figure above. It seems that it fluctuates around a mean value about 1.5. Ignoring some high jumps within the data, we may say that it converged. The trace plot seems that converged as it started and no obvious burn in part may be distinguished precisely.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(coda)
library(ggplot2)
target <- function(x){
  return(x**5 * exp(-x))
}

lnorm.HM = function(n, start.point){

  x = rep(0, 20000)
  x[1] = 1
  for (i in 2:length(x)) {
    x_cur = x[i-1]
    x_propos = rlnorm(1, meanlog = x_cur)
    A = min(1,(target(x_propos)*dlnorm(x_cur, meanlog = x_propos))/(target(x_cur)*dlnorm(x_propos, meanl

    if(runif(1) < A){
      x[i] = x_propos}else{
        x[i] = x[i-1]

      }
  }

  return(x)

}

set.seed(12345)
x.lnorm = lnorm.HM(10000, 1)

plot(x.lnorm, type = "l", main = "Time series of the generated distribution")
#hist(x.lnorm, breaks = 50, main = "Histogram of the generated distribution" )


chi.HM = function(n, start.point){

  x = rep(0, n)
  x[1] = start.point
  for (i in 2:length(x)) {
    x_cur = x[i-1]
    x_propos = rchisq(1, df=floor(x_cur+1))
    A = min(1,(target(x_propos)*dchisq(x_cur,floor(x_propos+1)))/(target(x_cur)*dchisq(x_propos,floor(x_
    if(runif(1) < A){
      x[i] = x_propos}else{
        x[i] = x[i-1]
      }
  }
  return(x)
}

set.seed(12345)
x.chi = chi.HM(10000, 1)
```

```r
plot(x.chi, type = "l", main = "Time series of the generated distribution")
#hist(x.chi, breaks = 50, main = "Histogram of the generated distribution" )
set.seed(12345)
Data = mcmc.list()

for (i in 1:10) {
  Data[[i]] <- as.mcmc(chi.HM(10000, i))

}
print(gelman.diag(Data))
gelman.plot(Data)

mean(x.lnorm)
mean(x.chi)
chemic=load("E:/LiU/2nd Semester/Computational Statistics/Labs/4/chemical.RData")
chemic = data.frame(Day = X, Concentration = Y)

ggplot(chemic, aes(Day, Concentration))+geom_point()+theme_bw()
y = chemic$Concentration
n= nrow(chemic)

M = 1000
Var = 0.2

mu = matrix(0, nrow = M, ncol = n)
set.seed(12345)
for (i in 1:M) {

  for (j in 1:n) {
    if(j ==1){
      mu[i,j] = rnorm(1, mean = (y[1]+mu[i,2])/2, sd = sqrt(Var/2))
      }else if(j == n){
        mu[i,j] = rnorm(1, mean = (y[n] + mu[i,n-1])/2, sd = sqrt(Var/2))
        }else{
          mu[i,j] = rnorm(1, mean = (y[j] + mu[i,j-1] + mu[i,j+1])/2,
                          sd = sqrt(Var/3))
        }
  }
}


Mu.mean = apply(mu, 2, mean)
chemic$Mu = Mu.mean

chemic$Type1 = "Observations"
chemic$Type2 = "Gibbs"


ggplot(chemic)+ geom_point(mapping = aes(Day, Concentration, color=Type1))+
  geom_line(mapping = aes(Day, Mu, color = Type2)) + theme(legend.position = "bottom")
plot(mu[,50], type = 'l', main = "Trace plot of mu_50")
```